

Reachability analysis of dynamical systems having piecewise-constant derivatives[☆]

Eugene Asarin^{a,1}, Oded Maler^{b,*}, Amir Pnueli^c

^a *Institute for Information Transmission Problems, 19 Ermolovoy st., Moscow, Russian Federation*

^b *SPECTRE-VERIMAG², Miniparc-ZIRST, 38330 Montbonnot, France*

^c *Department of Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel*

Abstract

In this paper we consider a class of hybrid systems, namely dynamical systems with piecewise-constant derivatives (PCD systems). Such systems consist of a partition of the Euclidean space into a finite set of polyhedral sets (*regions*). Within each region the dynamics is defined by a constant vector field, hence discrete transitions occur only on the boundaries between regions where the trajectories change their direction.

With respect to such systems we investigate the reachability question: *Given an effective description of the systems and of two polyhedral subsets P and Q of the state-space, is there a trajectory starting at some $x \in P$ and reaching some point in Q ?* Our main results are a decision procedure for two-dimensional systems, and an undecidability result for three or more dimensions.

1. Introduction

1.1. Motivation

Hybrid systems (HS) are systems that combine intercommunicating discrete and continuous components. Most embedded systems belong to this class since they operate and interact with a continuous environment, and are expected to provide real-time responses to continuously varying situations.

[☆] This research was supported in part by the France–Israel project for cooperation in Computer Science, by the European Community ESPRIT Basic Research Action Projects REACT (6021) and by Research Grant #01-00884 of the Russian Foundation of Fundamental Research.

* Corresponding author. E-mail: Oded.Maler@imag.fr.

¹ During the preparation of the paper the author was visiting the University of Paris 12, Val de Marne.

² VERIMAG is a joint laboratory of CNRS, INPG, UJF and VERILOG SA, SPECTRE is a project of INRIA.

The introduction of HS models is motivated by a real practical concern: with the decrease in the size and price of computing elements, more and more computers (discrete state-transition systems) are embedded within real-world control loops such as in avionics, process control, robotics and consumer products – to mention a few application areas. The analysis and prediction of the combined behavior of these embedded systems require formal tools that cut across existing disciplinary boundaries: the real-world is usually modeled by control engineers as a continuous dynamical system while computer scientists investigate the dynamics of discrete systems.

The ultimate goal of the theory of HS is to build models of such embedded systems, models which include the dynamics of an external environment and the interface between the controller and the environment. Within these models, based upon a description of a discrete controller (such as a program or a digital circuit) and upon its timing characteristics, it will be possible to prove that the behavior of the controlled environment satisfies certain properties. Even if we cannot realistically hope for fully algorithmic analysis techniques, any progress along this line of research will enhance the quality of current design methodologies, and will provide system developers with models and with software tools that will result in a more efficient, systematic and reliable development process.

1.2. Models for hybrid systems

Hybrid systems generalize both discrete state-transition systems and continuous dynamical systems. A HS consists of two types of state-variables: discrete variables whose values change via discrete state-transitions, and continuous variables which change continuously according to some dynamical law during the interval between two consecutive state-transitions. These two types of variables interact with each other in the following ways:

1. Some property satisfied by continuous variables (e.g., a variable crosses a threshold) enables or disables a discrete state-transition.
2. A change in a discrete variable may change the dynamical law to which some continuous variables are subject.

The first formal model in the verification literature linking continuous and discrete dynamics was the *phase-transition systems* introduced in [13]. Several, more or less similar, models for hybrid systems have been proposed and investigated recently (see e.g. [15, 2]). Various negative and positive results concerning the decidability of verification problems in these models have been established. The positive results usually involved the special case of timed automata [3, 1, 10], whose introduction was motivated by real-time systems. Timed automata can be viewed as hybrid systems where the only continuous variables are timers, all varying at the same rate, and possibly being reset by discrete transitions. Tests on values of those timers serve as “guards” for performing the discrete transitions – usually those tests are conjunctions of simple linear inequalities in one variable, or linear inequalities on the difference between two variables.

Similar to [12] (integration graphs), we are looking for decidable subclasses of hybrid systems with piecewise-constant derivatives (PCD systems). The strategy taken in [12] is based on placing restrictions on the guards of transitions that may occur within loops. Here we take an alternative approach and allow arbitrary boolean combinations of linear inequalities as transition guards, but place the following restrictions:

- The system is deterministic (this restriction is not essential for the results).
- Transitions do not modify any of the continuous variables. Thus, there are no resets or other assignment statements.
- The control component of a state is determined by the values of the continuous variables. Thus, if the corresponding PCD system is represented as a state-graph, it is not possible to reach different nodes (possibly in different computations) with the same set of values for the continuous variables.

Using the terminology of [15], the guards for all the transitions outgoing from a given location are mutually exclusive and their union is the complement of the invariant condition for that location.

Due to the fact that the discrete (control) component of the state is fully determined by the values of the continuous variables and that all changes are continuous, we prefer to present the system without explicit reference to discrete states. Instead, the system is viewed as a set of *regions* (corresponding to discrete states in other presentations) with boundaries separating them. Each region is associated with a constant *vector field* which identifies the rates at which the various variables change. Reaching a boundary and crossing into another region is equivalent to taking a transition to another discrete state in which the continuous evolution rule is different.

Compared to classes of hybrid and real-time systems considered so far in the verification literature, the model investigated in this paper is in some aspects, more general and in some aspects more restrictive. On one hand, we drop the restriction of a uniform slope for all continuous variables, assumed in time automata, and allow each variable to have its own slope within a discrete state. We also allow the continuous variables to appear in more general guards for discrete transitions (combinations of arbitrary linear equalities). On the other hand, our class of systems is more restrictive, mainly because we do not allow discrete “jumps” in the values of the variables (such as those caused by assignment statements). Thus the trajectories of the system are continuous, but not smooth. Systems obeying such requirement are closer to continuous dynamical systems and are more amenable to topological and geometrical analysis. Some of the recent research in hybrid systems may appear to be too dominated either by continuous or by computer science techniques, depending on the authors’ origin. The model underlying this paper can be seen as an attempt to balance the situation by generalizing the continuous dynamics and simplifying the discrete component.

1.3. An example: hunter and the hunted

In order to motivate the reader we will present a toy problem which is analyzable using the techniques developed in this paper.

The problem involves two players which move in an one-dimensional space. Their respective positions are denoted by the state-variables x and y and we assume that initially $x < y$. When x and y occupy certain positions, party y is the pursuer while party x flees away from y . However, when x and y occupy different positions, the roles may be reversed and x may assume the role of a pursuer while y flees away. Such a situation arises, for example, in the video game *Pacman* in which, under normal circumstances, the Pacman is pursued by one or more ghosts. However, when the Pacman eats a certain fruit, the roles are reversed and the Pacman starts chasing the ghost. We therefore may consider the position of the Pacman and its pursuing ghost to be given by x and y , respectively.

The behavior of each of the players consists of running either left or right at certain velocities depending on the relative location of their opponents. The ghost runs to the left at velocity b_1 when $x < 0$ and runs to the right at velocity b_2 when $x > 0$. The Pacman runs to the left at velocity a_1 when $y > 2$ and runs to the right at velocity a_2 when $y < 2$.

The configuration of the two players is displayed in Fig. 1. The precise behavioral rules of Pacman and the ghost are depicted in Table 1. All the parameters are positive and the entries in the table denote velocities.

Knowing all the parameters of the system we would like to answer questions such as: Given that Pacman starts at some position in the interval $[x_0, x_1]$, and that the ghost starts at some position in $[y_0, y_1]$, will they ever meet ($x = y$)? Is it possible that the distance between them will become larger than some d^* ? Will Pacman ever reach some point x^* ?

If we look at the positions of Pacman and the ghost as the coordinates of our system, we obtain a planer (two-dimensional) PCD system (see exact definitions below). Each point in the x, y plane represents the joint positions of Pacman and the

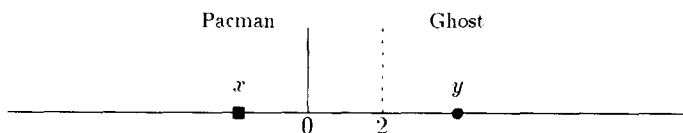


Fig. 1. Pacman x and the ghost y .

Table 1
The behavioral rules of Pacman and the ghost

	$x > 0, y > 2$	$x < 0, y > 2$	$x < 0, y < 2$	$x > 0, y < 2$
V_{Pacman}	$-a_1$	$-a_1$	a_2	a_2
V_{Ghost}	b_2	b_1	$-b_1$	b_2

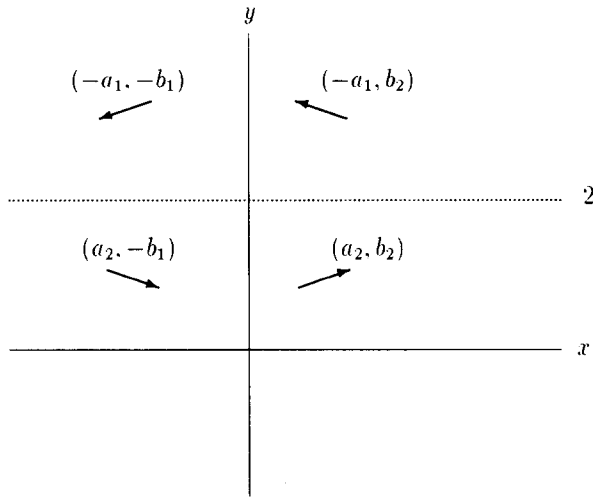


Fig. 2. Pacman x and the ghost y viewed as a planar PCD system.

ghost. Their corresponding rules of behavior induce a partition of the plane into regions such that within every region the system evolves with a constant slope of the form $c = (V_{\text{Pacman}}, V_{\text{Ghost}})$ as depicted in Fig. 2. In this paper we show how for every system of this type, reachability questions between polyhedral subsets of the state-space can be effectively answered. On the other hand, we show that for three-dimensional systems (e.g., by adding a third player to the game) there is no general reachability algorithm.

The rest of the paper is organized as follows. In Section 2 we introduce the necessary geometrical and topological prerequisites and define the class of systems we are dealing with. In Section 3 we present planar systems and prove some of their important properties which are crucial for the termination of our decision procedure. The computational framework for forward simulation is developed in Section 4, culminating in the decision procedure for reachability between points, which is extended in Section 5 to reachability between regions. In Section 6 we demonstrate the computational power of PCD systems and show that three dimensions are sufficient for simulating two-stack machines and, hence, the reachability problem for three-dimensional systems is undecidable.

The paper is a combination and elaboration of results presented in [14, 4].

2. Preliminaries

Throughout this paper, we deal with the d -dimensional Euclidean space¹ $X = \mathbb{R}^d$. Points (vectors) in X are denoted by boldface letters such as x or a . The expression

¹ A readable introduction to convex and polyhedral sets can be found in [7].

$\mathbf{a} \cdot \mathbf{x}$ denotes the standard inner product. For $\mathbf{x}_1, \dots, \mathbf{x}_n \in X$ an affine combination is $\mathbf{x} = \sum \lambda_i \mathbf{x}_i$ such that $\sum \lambda_i = 1$. A convex combination is the same with $\lambda_i > 0$ for every i . The affine (resp. convex) hull of a set $A \subseteq X$ is the set of all affine (resp. convex) combinations of points in A . They are denoted by $\text{aff}(A)$ and $\text{conv}(A)$. A subset P of X is convex if $P = \text{conv}(P)$. The dimension of P , $\dim(P)$ is the dimension of $\text{aff}(P)$. Intuitively, the affine hull is the generalization of concepts such as “the line connecting two points” or “the plane induced by three non-colinear points”, etc.

The interior $\text{int}(P)$ of $P \subseteq X$ is the set of points $\mathbf{x} \in P$ such that there exist a neighborhood $N_\epsilon(\mathbf{x}) \subseteq X$ such that $N_\epsilon(\mathbf{x}) \subseteq P$. The boundary of P is $\text{bd}(P) = \text{cl}(P) - \text{int}(P)$, where $\text{cl}(P)$ denotes closure. If P is e -dimensional for some $e < d$ then it has no interior points because every neighborhood in X^d is d -dimensional and thus contains parts outside P . On the other hand, we can define the notion of relative-interior by relaxing the above condition into $(N_\epsilon(\mathbf{x}) \cap \text{aff}(P)) \subseteq P$. The set of all relative interior points of P is denoted by $\text{ri}(P)$. Similarly the relative boundary of P is defined as $\text{rb}(P) = \text{cl}(P) - \text{ri}(P)$. For example a closed segment in a two-dimensional space has no interior points, but its “open” subsegment is its relative interior and its endpoints constitute its relative boundary.

An open (closed) *half-space* in X is the set of all points $\mathbf{x} \in X$ satisfying $\mathbf{a} \cdot \mathbf{x} + b < 0$ ($\mathbf{a} \cdot \mathbf{x} + b \leq 0$). A *convex polyhedral set* is an intersection of finitely many half-spaces. A *time segment* is any interval $[0, r] \subseteq \mathbb{R}^+$ including \mathbb{R}^+ itself. A *trajectory* in X is a continuous function $\xi: T \rightarrow X$, where T is a time segment.

Definition 1 (*Dynamical system*). An (autonomous) dynamical system is $\mathcal{H} = (X, f)$ where X is the state-space and f is a partial function from X to X such that $d^+ \mathbf{x}/dt = f(\mathbf{x})$ is the differential equation governing the evolution of \mathbf{x} . A trajectory of \mathcal{H} starting at some $\mathbf{x}_0 \in X$ is $\xi: T \rightarrow X$ such that ξ is a solution of the equation with initial condition $\mathbf{x} = \mathbf{x}_0$, i.e., $\xi(0) = \mathbf{x}_0$ and for every t , $f(\xi(t))$ is defined and is equal to the right derivative of $\xi(t)$.

A differential equation has a uniqueness property if for every \mathbf{x}_0 there is at most one solution. In this case the system is said to be deterministic. This paper treats a subclass of dynamical systems, namely those having piecewise-constant (possibly discontinuous) derivatives:

Definition 2 (*PCD system*). A piecewise-constant derivative (PCD) system is a dynamical system $\mathcal{H} = (X, f)$, where f is a (possibly partial) function from X to X such that the range of f is a finite set of vectors $C \subset X$, and for every $\mathbf{c} \in C$, $f^{-1}(\mathbf{c})$ is a finite union of convex polyhedral sets.

In other words, a PCD system consists of partitioning the space into convex polyhedral sets (“regions”), and assigning a constant derivative \mathbf{c} (“slope”) to all the points sharing the same region. The trajectories of such systems are broken lines, with the breakpoints occurring on the boundaries of the regions. The example in the introduction (Fig. 2) is a PCD system.

A description of a PCD system is simply a list of the regions (expressed as intersections of linear inequalities) and their corresponding slope vectors. From now on we assume that all the constants in the system's definition are rational. Note that, unlike more general dynamical systems, PCD systems are effective in the following sense: Given a description of the system, and a rational initial point x , there exists some positive $\varepsilon > 0$ such that, for every Δt , $0 < \Delta t < \varepsilon$, one can calculate *precisely* the point x' which a trajectory starting at x will reach after time Δt .

Given a description of a PCD system \mathcal{H} , the reachability problem for \mathcal{H} , denoted by **Reach**(\mathcal{H}, x, x') is the following: *Given $x, x' \in X$, are there a trajectory ξ and $t \geq 0$ such that $\xi(0) = x$ and $\xi(t) = x'$?* The region-to-region reachability problem **R-Reach**(\mathcal{H}, P, P') is: *Given two polyhedral sets $P, P' \subseteq X$, are there two points $x \in P$ and $x' \in P'$ such that the answer to **Reach**(\mathcal{H}, x, x') is positive?*

3. Planar systems and their properties

In the following sections we will concentrate on planar PCDs, i.e., $X = \mathbb{R}^2$, with the additional restriction that all the regions are two-dimensional. For every vector $x = (x_1, x_2) \in \mathbb{R}^2$ we define its *right rotation* as the vector $\hat{x} = (x_2, -x_1)$. Clearly $x \cdot \hat{x} = 0$ and $x \cdot y = \hat{x} \cdot \hat{y}$.

Definition 3 (Polyhedral partition). A finite polyhedral partition of X is a family $\mathcal{P} = \{P_1, \dots, P_k\}$ of open full-dimensional polyhedral sets such that $\bigcup_{i=1}^k \text{cl}(P_i) = X$ and for every $P_i, P_j \in \mathcal{P}$, $P_i \cap P_j = \emptyset$.

We will denote by $bd(\mathcal{P})$ the set of all points in X which are in $bd(P)$ for some $P \in \mathcal{P}$ and by $E(\mathcal{P})$ the set of *edges* of \mathcal{P} , namely nonempty subsets of X of the form $e = \text{ri}(\text{cl}(P_i) \cap \text{cl}(P_j))$ for some $P_i, P_j \in \mathcal{P}$. Similarly the set of *vertices* of \mathcal{P} , $V(\mathcal{P})$ consists of points $x \in X$ such that $\{x\} = \text{cl}(e_i) \cap \text{cl}(e_j)$ for some $e_i, e_j \in E(\mathcal{P})$. We call the elements of $B(\mathcal{P}) = E(\mathcal{P}) \cup V(\mathcal{P})$ *boundary elements*. One can easily see that X is decomposed into a disjoint union $\mathcal{P} \cup E(\mathcal{P}) \cup V(\mathcal{P})$. For example, the polyhedral partition in Fig. 3 has 5 regions, 3 vertices and 7 edges.

Suppose e is an edge such that $e \subseteq bd(P) \cap bd(P')$. Let $P = \{x: \bigwedge_{i \in I} a_i \cdot x + b_i < 0\}$ and $P' = \{x: \bigwedge_{i' \in I'} a_{i'} \cdot x + b_{i'} < 0\}$. Then for e to be nonempty there must be some $j \in I, k \in I'$, such that $a_j = -a_k$ and $b_j = -b_k$ and every $x \in e$ satisfies $a_j \cdot x + b_j = 0$. We call a_j and a_k the *characteristic vectors* of e relative to P and P' . One can see that they are two opposite normals to e .

Definition 4 (Planar PCD systems). A planar PCD system is given by $\mathcal{H} = (\mathcal{P}, \varphi, \psi)$, where \mathcal{P} is a polyhedral partition of $X = \mathbb{R}^2$, $\varphi: \mathcal{P} \rightarrow X$ is a function which assigns to each region a slope vector in X , and a function $\psi: B(\mathcal{P}) \rightarrow \mathcal{P}$ satisfying $b \subseteq \text{cl}(\psi(b))$ for every boundary element (edge or vertex) $b \in B(\mathcal{P})$.

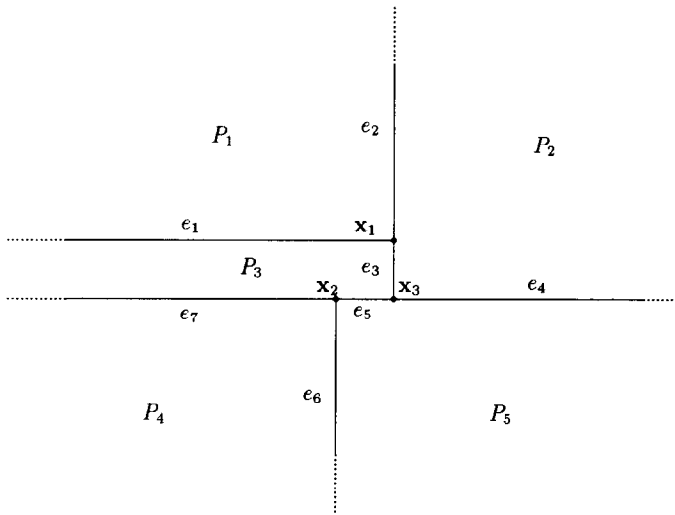


Fig. 3. A polyhedral partition of the plane.

The function ψ simply associates every boundary element with one of its neighboring regions. One can easily see that by letting $f(x) = \varphi(P)$ when $x \in P \in \mathcal{P}$ and $f(x) = \varphi(\psi(b))$ when $x \in b \in B(\mathcal{P})$ we obtain the more general definition 2. We denote $\varphi(P_i)$ by c_i .

Note that we may have an edge e and a vertex $x \in cl(e)$ such that $\psi(e) \neq \psi(x)$. For example, in the partition of Fig. 3, we may have $\psi(e_1) = P_3$, while $\psi(x_1) = P_2$.

3.1. Orientation and ordering of boundaries

For $b \in B(\mathcal{P})$ such that $\varphi(P) = c$, we say that b is an *entry boundary element* of P if for every $x \in b$, we have $\{x + ct : 0 < t < \varepsilon\} \subseteq \text{int}(P)$ for some $\varepsilon > 0$. This implies that the vector c taken at any entry boundary point of P points into the interior of P . Similarly b is an *exit element* if the same condition holds for some $\varepsilon < 0$ and for all t , $\varepsilon < t < 0$.

Consider a region P and one of its edges e with a characteristic vector a relative to P (i.e., a points from e into P), and let $c = \varphi(P)$. Then, by simple calculation one can see that e is an *entry* to P iff $a \cdot c > 0$, e is an *exit* from P iff $a \cdot c < 0$, and e is *neutral* (neither entry nor exit) if $a \cdot c = 0$. For simplicity of presentation we assume that the system is not degenerate, i.e., no edge is neutral with respect to a neighboring region. If e is on the boundaries of P_i and P_j , it is required that e be an entry to one of them (say P_i) and an exit to the other, and be affiliated with the region to which it is an entry, i.e., $\psi(e) = P_i$. From now on, we adopt the convention that the characteristic vector of an edge e is the one pointing into the region $P = \psi(e)$, i.e., the region to which e is an entry edge.

For every vertex $x \in bd(P)$ there are exactly two edges $e, e' \subseteq bd(P)$ such that $x = cl(e) \cap cl(e')$. Then one can see that x is an entry point to P if both e and e' are

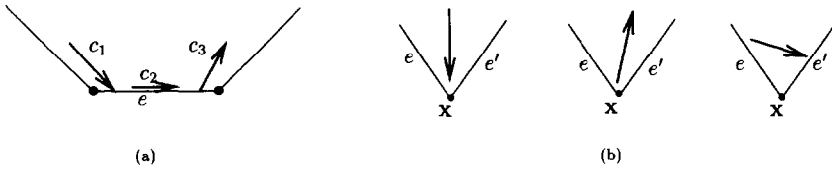


Fig. 4. The possible orientations between a region and its borders: (a) e is an exit, a tangent or an entry to P according to whether $\varphi(P)$ is c_1 , c_2 or c_3 respectively (b) x is an exit, an entry, or neutral with respect to P .

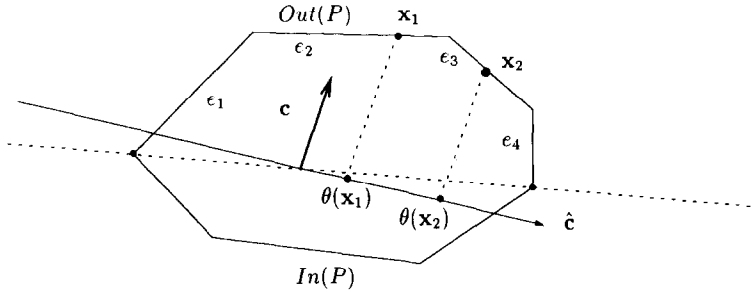


Fig. 5. Ordering on the boundary: $x_1 \leq x_2$.

entry edges. Symmetrically it is an exit point if both e and e' are exit edges. Otherwise, when one of $\{e, e'\}$ is an entry edge and the other is an exit edge we say that x is neutral w.r.t. P . We require that $\psi(x) = P$ iff x is an entry point of P . Thus we have completed the classification of all boundary points according to their orientation relative to the region (see Fig. 4). We will denote the set of all entry points of P by $In(P)$ and set of all exit points by $Out(P)$.

Next we prove some fundamental properties of planar systems which apply to an even more general class than the PCD systems considered in this paper. A sufficient condition for these properties to hold is that any straight line can be divided into finitely many segments, each of which can be traversed by any trajectory in at most one direction.

Consider a region P with $\varphi(P) = c$ whose boundary is partitioned into $In(P)$ and $Out(P)$. The mapping $\theta: X \rightarrow \mathbb{R}$, defined as $\theta(x) = x \cdot \hat{c}$, assigns to every $x \in X$ a value proportional to the length of the projection of the vector x on the right rotation of c . One can easily see that the relation \leq , defined as $x_1 \leq x_2$ if $\theta(x_1) \leq \theta(x_2)$, is a dense linear order on $In(P)$ and $Out(P)$ (see Fig. 5).

The fact that $In(P)$ and $Out(P)$ are ordered allows us to speak of *boundary intervals* of the form $[x_1, x_2]$ denoting all the points $x \in In(P)$ (or $Out(P)$) satisfying $x_1 \leq x \leq x_2$. We use $<$ to denote the strict variant of \leq and say that $e_1 < e_2$ if $x_1 < x_2$ for every $x_1 \in e_1, x_2 \in e_2$. For example, in Fig. 5 we have $e_1 < e_2 < e_3 < e_4$.

Claim 1 (Fundamental property of planar systems). *Let ξ be any trajectory that intersects $In(P)$ (or $Out(P)$) in three consecutive points, x_1 , x_2 and x_3 . Then, $x_1 \leq x_2$ implies $x_2 \leq x_3$ and $x_1 \geq x_2$ implies $x_2 \geq x_3$.*

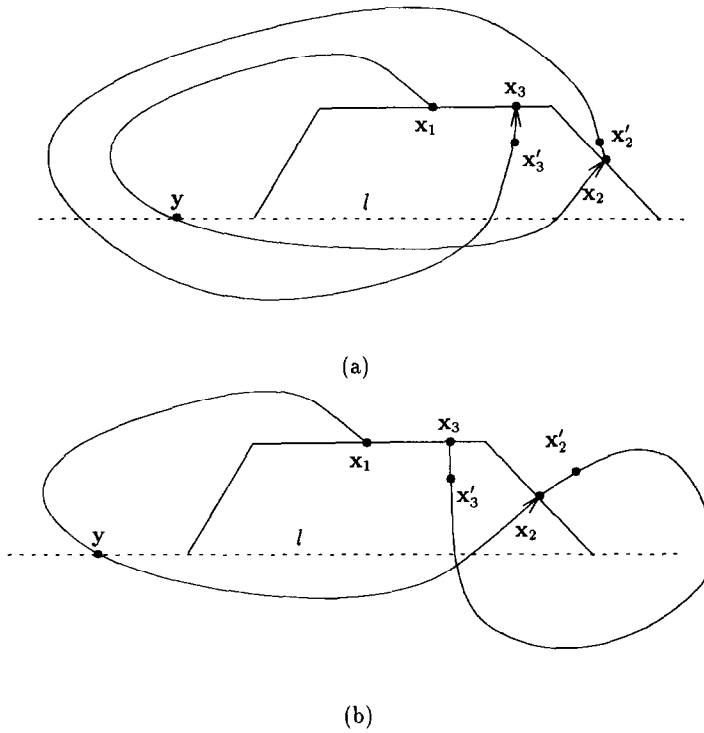


Fig. 6. Illustration of the fundamental property: (a) Circumvention from the left. (b) Circumvention from the right.

Proof. If $x_1 = x_2$ then due to determinism $x_2 = x_3$. So we assume $x_1 < x_2$ and show that $x_3 < x_2$ implies an intersection of ξ with itself in the segment between x_2 and x_3 , which contradicts $x_3 < x_2$. Let l be some line separating $In(P)$ and $Out(P)$. Suppose, without loss of generality, that the trajectory from x_1 to x_2 circumvents P from the left (i.e., its last intersection with line l before re-entering P is some $y < x_1$) (see Fig. 6). Consider now the closed set S bounded by the closed curve consisting of the trajectory from x_1 to x_2 and the boundary interval $[x_1, x_2]$. In order that there will be a trajectory from x_2 to x_3 there must be two points, x'_2 “above” x_2 and x'_3 “below” x_3 such that there is a trajectory $x_2 \rightarrow x'_2 \rightarrow x'_3 \rightarrow x_3$ and in particular we can choose x'_2 outside S and x'_3 inside S (if we cannot, then x_2 and x_3 coincide). Consequently, according to Jordan’s theorem, the trajectory $x'_2 \rightarrow x'_3$ must intersect the boundary of S . Since it cannot do it on $[x_1, x_2]$, ξ must intersect itself and this contradicts determinism unless $x_2 = x_3$. This is true independent of whether the trajectory from x_2 to x_3 circumvents P from the left (Fig. 6(a)), or from the right (Fig. 6(b)). \square

This topological property has many consequences concerning the set of possible trajectories. It implies that the sequence of consecutive intersection points of a trajectory with $In(P)$ or $Out(P)$ is monotone with respect to \preceq . In fact, the relation between

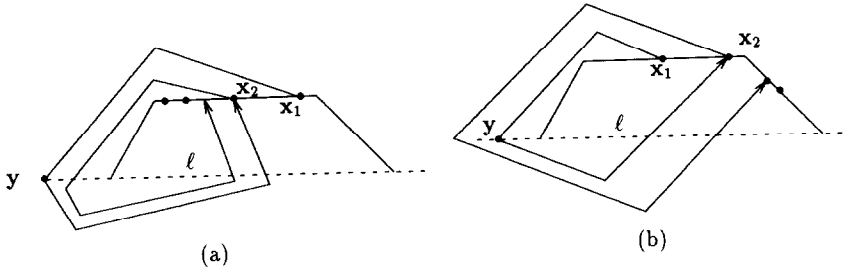


Fig. 7. (a) A contracting spiral. (b) An expanding spiral.

x_1 , x_2 and y determines two classes of possible “quasi-cycles”: if x_2 is between x_1 and y we have a “contracting spiral” (Fig 7(a)) while if x_1 is between x_2 and y we have an “expanding spiral” (Fig. 7(b)). Once a trajectory has performed a contracting spiral, it will never reach any point “outside” the spiral (i.e., outside the closed curve formed by the trajectory $x_1 \rightarrow y \rightarrow x_2$ and the boundary interval $[x_1, x_2]$) and, symmetrically, after performing an expanding spiral, a trajectory will never reach a point “inside” the spiral.

3.2. Traces and signatures

Let $\xi: T \rightarrow X$ be trajectory defined on a polyhedral partition \mathcal{P} . The *trace* of ξ is the sequence $\bar{\xi} = x_1, x_2, \dots$ of the intersection points of ξ with $B(\mathcal{P})$. We use the notation $\bar{\xi}_{[i..j]}$ to denote the subsequence x_i, \dots, x_j of $\bar{\xi}$. We say that x_{i+1} is the *immediate successor* of x_i .

Note that the finiteness of $\bar{\xi}$ is not necessarily determined by the boundedness of the time interval T : a trajectory ξ defined over the whole \mathbb{R}^+ may reach some “terminal” region and hence will stop crossing boundaries and $\bar{\xi}$ will be a finite sequence. On the other hand, a contracting spiral can cross boundaries infinitely many times during a finite real-time interval.

The qualitative behavior of the trajectory ξ such that $\bar{\xi} = x_1, x_2, \dots$, is captured by its *signature*, which is the sequence of boundary elements (edges and vertices), $\tau(\xi) = b_1, b_2, \dots$ such that for every i , $x_i \in b_i$. The sequence $\rho(\xi) = P_1, P_2, \dots$ is the corresponding *region signature* of ξ satisfying the obvious relation $b_i \subseteq \text{In}(P_i)$. Since vertices can be seen as a degenerate case of edges, we will henceforth refer to signatures as to sequences of *edges*.

A sequence $w = w_1, \dots, w_n$ over some finite alphabet is called a (primitive) *cycle* if $w_1 = w_n$ and $\{w_i, \dots, w_{n-1}\}$ are pairwise different. A trace $\bar{\xi}_{[i..j]} = x_i, x_{i+1}, \dots, x_j$ forms a *region cycle* if its corresponding region signature, P_i, \dots, P_j is a cycle. It forms an *edge cycle* if its signature e_i, \dots, e_j is a cycle. Note that if $\bar{\xi}_{[i..j]}$ is an edge cycle it is also a region cycle but not vice versa.

An edge e is said to be *abandoned* by a trajectory after position i , if $e_i = e$ and for some $j, k, i \leq j < k$, $\bar{\xi}_{[j..k]}$ forms a region cycle and $e \notin \{e_{i+1}, \dots, e_k\}$.

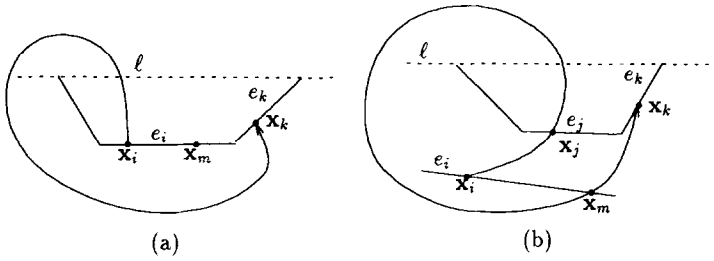


Fig. 8. Irreversibility of abandonment.

Claim 2 (Abandonment is irreversible). *An edge e abandoned after position i will not appear in the signature at any position $m > i$.*

Proof. Let $\bar{\xi}_{[j..k]}$ be the trace that forms the relevant region cycle. We assume, without loss of generality, that $x_j < x_k$ and that the cycle circumvents P_j from the left (expanding spiral). Consider the case of $i = j$. Then, another intersection of ξ with e_i must happen at some $x_m < x_k$ and this will violate the monotonicity property (see Fig. 8(a)). For the case $i < j$, e_i is an edge not belonging to $In(P_j)$ and x_i must be inside the set enclosed by the trajectory from x_j to x_k . If e_i is completely contained within this set, it cannot be reached from x_k without self-intersection of ξ . If e_i is not fully contained, then it must intersect the trajectory from x_j to x_k and will appear in the signature at some position m , $j < m < k$ and contradict the assumption of abandonment (see Fig. 8(b)). A similar argument holds for the case of a contracting spiral. \square

Corollary 3. *Every trajectory has an ultimately periodic signature, i.e., a signature of the form $e_1, \dots, e_i, (e_{i+1}, \dots, e_{i+j})^\omega$ for some finite i, j , where j is at most the number of regions.*

Proof. Because of the finite number of edges every infinite signature contains a subset E of edges appearing infinitely often. Due to claim 1, this subset may contain at most one entry edge per region. Let i be a position in the signature after all elements of E have already occurred and all the elements of $E(\mathcal{P}) - E$ have already disappeared. Let $e = e_i$, then the remaining signature can be factorized into $e, \sigma_1, e, \sigma_2, e \dots$ where $\sigma_j \in (E - \{e\})^*$ for every $j > 0$. All the elements of $E - \{e\}$ must appear in every σ_j , otherwise they are abandoned. In addition every σ_j is cycle-free, otherwise e must be abandoned. Finally, it is impossible for an edge e' to appear before e'' in σ_j and after e'' in σ_{j+1} as in the sequence

$$e \dots e' \dots, e'' \dots, e \dots, e'', \dots, e', \dots, e$$

because otherwise e' will be abandoned due to the cycle $e'' \dots, e \dots, e''$. Consequently, all the σ_j must be identical and the sequence is ultimately periodic. \square

Equipped with this nice qualitative property (which does not hold in higher dimensions), we turn to the actual calculation of trajectories of a given PCD system.

4. Point-to-point reachability

In this section we devise a framework for representing edges and vertices that will provide for the exact characterization and calculation of the set of points reachable by a trajectory starting at a given point.

Definition 5 (Representations). Let $e \in \text{In}(P)$ be an edge with a characteristic vector \mathbf{a} (pointing into P). A representation for e consists of two vectors $\mathbf{v} \in \text{cl}(e)$, $\mathbf{u} = \hat{\mathbf{a}}$ (the right rotation of \mathbf{a}), and two numbers l, h , such that l is a rational or the special value $-\infty$, $h > l$ is a rational or ∞ , and

$$e = \{\mathbf{v} + \lambda \mathbf{u} : l < \lambda < h\}.$$

The choice of \mathbf{v} implies, of course, the values of l and h . Having fixed \mathbf{v} and \mathbf{u} for every edge we can uniquely represent every nonvertex boundary point \mathbf{x} by a pair (e, λ) identifying the edge e and the parameter λ . Every open boundary interval $(\mathbf{x}_1, \mathbf{x}_2)$ contained in e is represented by some *edge interval* $(e, (\lambda_1, \lambda_2))$, $l \leq \lambda_1 \leq \lambda_2 \leq h$. Vertices are represented by themselves.

Definition 6 (Successor function). Let e and e' be two edges with (\mathbf{u}, \mathbf{v}) - and $(\mathbf{u}', \mathbf{v}')$ -representations. The partial function $f_{e,e'} : (l, h) \rightarrow (l', h')$ is defined as follows: $f_{e,e'}(\lambda) = \lambda'$ iff $\mathbf{x}' = (e', \lambda')$ is the immediate successor of $\mathbf{x} = (e, \lambda)$.

Claim 4. Given a representation, the successor function is well-defined and computable.

Proof. Obviously if there is no P such that $e \subseteq \text{In}(P)$ and $e' \subseteq \text{Out}(P)$ then $f_{e,e'}$ is nowhere defined. Consider now $(e, \lambda) \in \text{In}(P)$ and its successor $(e', \lambda') \in \text{Out}(P)$ for some region P having a slope c . For (e', λ') to be indeed the successor of (e, λ) , the following vector equation must be satisfied for some $t > 0$:

$$\mathbf{v}' + \lambda' \mathbf{u}' = \mathbf{v} + \lambda \mathbf{u} + t\mathbf{c}$$

by rearranging, we obtain

$$\lambda' \mathbf{u}' = \lambda \mathbf{u} + (\mathbf{v} - \mathbf{v}') + t\mathbf{c}.$$

Multiplying both sides by $\hat{\mathbf{c}}$, the right rotation of \mathbf{c} , and observing that $\hat{\mathbf{c}} \cdot \mathbf{c} = 0$, we obtain

$$\lambda'(\hat{\mathbf{c}} \cdot \mathbf{u}') = \lambda(\hat{\mathbf{c}} \cdot \mathbf{u}) + \hat{\mathbf{c}} \cdot (\mathbf{v} - \mathbf{v}')$$

from which we get

$$\lambda' = A_{e,e'} \lambda + B_{e,e'},$$

where

$$A_{e,e'} = \frac{\mathbf{c} \cdot \mathbf{a}}{\mathbf{c} \cdot \mathbf{a}'} \quad \text{and} \quad B_{e,e'} = \frac{\hat{\mathbf{c}} \cdot (\mathbf{v} - \mathbf{v}')}{\mathbf{c} \cdot \mathbf{a}'}.$$

To obtain these expressions for $A_{e,e'}$ and $B_{e,e'}$, observe that $\hat{c} \cdot u = \hat{c} \cdot \hat{a} = c \cdot a$ and similarly, $\hat{c} \cdot u' = c \cdot a'$.

By the assumption of nondegeneracy, c is not parallel to e' and therefore the denominator is never zero. If λ' is outside the interval (l', h') then the trajectory starting at (e, λ) intersects another border element and $f_{e,e'}(\lambda)$ is undefined. \square

Similarly we can define a *predecessor* function and compute it by

$$\lambda = \frac{\lambda' - B_{e,e'}}{A_{e,e'}}.$$

Claim 5. If $e \in \text{In}(P)$ and $e' \in \text{Out}(P)$ then $A_{e,e'} > 0$.

Proof. Since e is an entry edge of P , $c \cdot a > 0$, where a is the characteristic vector of e which, according to our convention, points into P . As e' is not an entry edge of P , its characteristic vector a' points away from P , and the normal to e pointing into P is given by $-a'$. Since $e' \in \text{Out}(P)$, $c \cdot (-a') < 0$, leading to $c \cdot a' > 0$. It follows that $A_{e,e'} = c \cdot a / c \cdot a' > 0$. \square

The one-step successor function can be generalized naturally to signatures.

Definition 7 (*Signature successor function*). Let $\sigma = e_1, \dots, e_k$ be a signature. The signature successor is a partial function $f_\sigma: (l_1, h_1) \rightarrow (l_k, h_k)$ such that $\lambda_k = f_\sigma(\lambda_1)$ iff a trace x_1, \dots, x_k with $x_1 = (e_1, \lambda_1)$ has the signature σ and $x_k = (e_k, \lambda_k)$. We denote the interval on which f_σ is defined by $\text{dom}(f_\sigma)$.

It can be easily verified that f_σ can be computed from $f_{e_i, e_{i+1}}, i = 1 \dots k-1$, yielding

$$f_\sigma(\lambda_1) = A_\sigma \lambda_1 + B_\sigma,$$

with

$$A_\sigma = A_{e_1, e_2} \cdot A_{e_2, e_3} \dots A_{e_{k-1}, e_k}$$

and

$$B_\sigma = (\dots((B_{e_1, e_2} \cdot A_{e_2, e_3} + B_{e_2, e_3}) \cdot A_{e_3, e_4} + B_{e_3, e_4}) \dots) \cdot A_{e_{k-1}, e_k} + B_{e_{k-1}, e_k}.$$

Of particular interest are the successor functions associated with cyclic signatures $\sigma = e_1, \dots, e_k$, where $e_k = e_1$.

Suppose $\xi = x_1, x_2, \dots$ with $x_i = (e_i, \lambda_i)$ for every $i \in \{1 \dots k\}$, has a periodic signature $(e_1, \dots, e_{k-1})^\omega$ and let $\sigma = e_1, \dots, e_k$. Then the sequence of intersection points of ξ with e_1 is represented by $(e_1, \mu_0), (e_1, \mu_1), \dots$ with $\mu_0 = \lambda_1$ and $\mu_{i+1} = A_\sigma \cdot \mu_i + B_\sigma$. It is straightforward to solve this linear difference equation and obtain the following expression for μ_n :

$$\mu_n = \begin{cases} \mu_0 + B_\sigma \cdot n & \text{if } A_\sigma = 1, \\ \mu_0 \cdot A_\sigma^n + B_\sigma \cdot (A_\sigma^n - 1) / (A_\sigma - 1) & \text{otherwise.} \end{cases}$$

Table 2
The limits of the intersection sequence
 $\mu_{n+1} = A_\sigma \mu_n + B_\sigma$ as a function of A_σ ,
 B_σ and μ_0 .

Case	Limit
$A_\sigma = 1, B_\sigma = 0$	μ_0
$A_\sigma = 1, B_\sigma > 0$	∞
$A_\sigma = 1, B_\sigma < 0$	$-\infty$
$A_\sigma < 1$	$\frac{B_\sigma}{1 - A_\sigma}$
$A_\sigma > 1, \mu_0 = \frac{B_\sigma}{1 - A_\sigma}$	μ_0
$A_\sigma > 1, \mu_0 > \frac{B_\sigma}{1 - A_\sigma}$	∞
$A_\sigma > 1, \mu_0 < \frac{B_\sigma}{1 - A_\sigma}$	$-\infty$

We can compute the limit μ^* of μ_n as n tends to infinity. The possible cases are listed in Table 2.

For every $e_i, i \in \{1, \dots, k-1\}$ in the signature we define its limit point λ_i^* by letting $\lambda_1^* = \mu^*$ and $\lambda_i^* = f_{e_{i-1}, e_i}(\lambda_{i-1}^*)$. Clearly the signature e_1, \dots, e_{k-1} can repeat forever only if for every $i, i \in \{1, \dots, k-1\}$ all its intersection points with the edge e_i are contained in the interval $(e_i, (l_i, h_i))$. We can now formulate a criterion for a cycle being repeated infinitely many times (see Fig. 9).

Criterion 1 (Infinity test). A trajectory starting with the cycle $(e_1, \lambda_1), \dots, (e_k, \lambda_k)$ such that $e_1 = e_k$, has a periodic signature $(e_1, \dots, e_{k-1})^\omega$ iff

$$l_i \leq \lambda_i^* \leq h_i \quad \text{for every } i \in \{1, \dots, k-1\}.$$

This criterion is based on the first occurrence of the cycle and enables us to decide in advance whether this cycle will repeat forever or some of its edges will be abandoned after finitely many iterations.

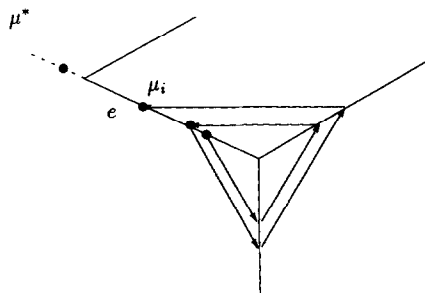


Fig. 9. A non-ultimate spiral: the limit of the sequence of intersection points with e is beyond the endpoint.

Theorem 6 (Point-to-point reachability). *The problem $\text{Reach}(\mathcal{R}, x, x')$ is decidable.*

Proof. We first show that it is decidable when x and x' are boundary points, $x = (e, \lambda)$ and $x' = (e', \lambda')$. All we need to calculate the successors and compare them with x' . During the generation of the trajectory we keep track of the nonabandoned edges (finitely many). When a cycle is detected the infinity test tells us whether or not this is the terminal cycle. If it is not we continue generating the successors. If it is the terminal cycle we can check whether e' is not in the cyclic signature or λ' is beyond the limit – in that case the answer is negative. Otherwise, after finitely many iterations we either reach x' or bypass it. This also implies that the problem is solvable for arbitrary points, because it is straightforward to compute the forward and backward intersections of trajectories passing through an interior point of a region with the boundaries. \square

If the target λ' is *exactly* the limit of the sequence, the reachability problem becomes an instantiation of Zeno's paradox, and the answer is a matter of definition.

5. Reachability between edges

The results so far allow us to compute the set of all points reachable by a single trajectory. Now we will show how reasoning about a noncountable number of trajectories departing from an edge interval can be reduced to reasoning about a finite number of trajectories.

5.1. Successor-trees

Claim 7. *Let (e', l') and (e', h') be the successors of (e, l) and (e, h) , respectively. Then for every λ , $l < \lambda < h$ the successor of (e, λ) is some (e', λ') , $l' < \lambda' < h'$.*

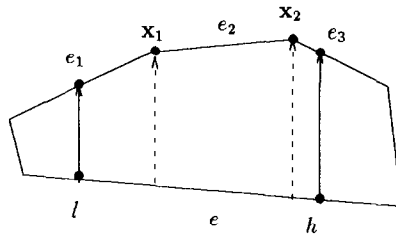
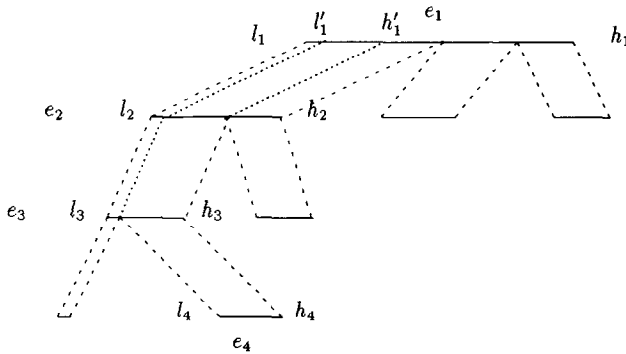
Proof. Follows from the monotonicity of the successor function and the convexity of the regions. \square

Consequently, for any edge interval $(e, (l, h))$, if the trajectories starting from (e, l) and (e, h) have identical signatures, so does every trajectory starting at (e, λ) , for some $l < \lambda < h$.

Next, we generalize the notion of a successor from single points to intervals.

Definition 8 (Edge-successors set). Let $(e, (l, h)) \subseteq \text{In}(P)$ be an edge interval. The successor set of e is a set of edges and vertices

$$\text{Succ}(e, (l, h)) = \{(e_1, (l_1, h_1)), x_1, (e_2, (l_2, h_2)), x_2, \dots, (e_n, (l_n, h_n))\},$$

Fig. 10. The successors of $(e, (l, h))$.Fig. 11. A finite portion of a successor-tree with a chain $(e_1, (l_1, h_1))$, $(e_2, (l_2, h_2))$, $(e_3, (l_3, h_3))$, $(e_4, (l_4, h_4))$. All the trajectories starting at some point in $(e_1, (l_1, h_1))$ have the same signature e_1, e_2, e_3, e_4 .

such that the elements are mutually disjoint, and their union is a connected boundary interval being in one-to-one correspondence with $(e, (l, h))$ via the successor/predecessor functions (see Fig. 10).

Claim 8. *The set of edge-successors is finite and computable.*

Proof. Because of the computability of the single point successor and predecessor functions. Moreover, for every i , all the points in $(e_i, (l_i, h_i))$ are the f_{e, e_i} -images of some subinterval of $(e, (l, h))$. \square

Clearly the set of all points in $Out(P)$ which are immediate successors of points in $(e, (l, u))$ is exactly the union of all elements in $Succ(e, (l, u))$.

Definition 9 (Successor-tree). A successor-tree rooted at $(e, (l, h))$ is constructed by calculating recursively the successors of every node starting at the root (see Fig. 11). A path along the tree is called a successor-chain.²

² For the sake of simplicity we ignore here the simpler case where one of the successors is a vertex – in this case a successor chain is simply a trace.

The notion of signature is generalized naturally to successor chains. If a chain has a signature σ then there must be at least one trajectory having this signature. Hence every infinite successor-chain must have an ultimately periodic signature.

The idea behind the edge-to-edge reachability algorithm is that all trajectories starting in some edge interval $(e, (l, h))$ and having the same signature are characterized by the behavior of two trajectories: the “leftmost” trajectory starting at l and the “rightmost” trajectory starting at h . By this we mean that

1. For trajectories with a finite signature $\sigma = e, \dots, e'$, the existence of a trajectory from some $x \in (e, (l, h))$ to some $x' \in (e', (l', h'))$ is exactly the existence of a nonempty intersection between the intervals (l', h') and $(f_\sigma(l), f_\sigma(h))$.
2. The criterion for a given cyclic signature to repeat forever is based on the convergence of the sequences associated with l and h .

For an edge interval $(e, (l, h))$ and a cyclic signature $\sigma = e, \dots, e$, we define

$$f_\sigma((l, h)) = \{\lambda' : \lambda' = f_\sigma(\lambda) \text{ for some } \lambda \in (l, h)\}.$$

Claim 9 (Decidability for periodic chains). *Consider the cyclic successor-chain $(e_1, (l_1, h_1)), \dots, (e_k, (l_k, h_k))$, $(e_1 = e_k)$, and let $\sigma = e_1, \dots, e_{k-1}$. It is decidable whether this finite successor-chain can be extended to an infinite periodic chain with the signature σ^ω , and if it is periodically extensible, whether there exists a trajectory among all those which share the same chain, reaching a point in an edge-interval $(e', (l', h'))$.*

Proof. Consider the sequence of intervals $(\mu_0, \eta_0), (\mu_1, \eta_1), \dots$ defined by $(\mu_0, \eta_0) = (l_k, h_k)$ and $(\mu_{n+1}, \eta_{n+1}) = (A_\sigma \mu_n + B_\sigma, A_\sigma \eta_n + B_\sigma)$. In Table 3, we present the different cases that may arise. For each of them, we specify the conditions for infinite periodic extension of the cyclic successor-chain and the set of points on edge e_1 which are reachable by such an extension if it exists. If no infinite extension exists, the third column is irrelevant. In this table, λ^* denotes $B_\sigma/(1 - A_\sigma)$, and we assume that e_1 is represented by the interval (L, H) . Cases a2–c2 are illustrated in Fig. 12.

Note that the condition $(\mu_0, \infty) \subseteq \text{dom}(f_\sigma)$ implies that all the edges in σ are unbounded from the right.

Table 3
Extensibility conditions and reachability sets

Case	Extensibility condition	e_1 -reachability set
a1. $A_\sigma = 1, B_\sigma = 0$	Always	(μ_0, η_0)
a2. $A_\sigma = 1, B_\sigma > 0$	$(\mu_0, \infty) \subseteq \text{dom}(f_\sigma)$	$\bigcup_{n \geq 0} (\mu_n, \eta_n)$
b1. $A_\sigma < 1, \lambda^* = \eta_0$	Always	(μ_0, η_0)
b2. $A_\sigma < 1, \eta_0 < \lambda^*$	$\lambda^* \in \text{dom}(f_\sigma)$	$\bigcup_{n \geq 0} (\mu_n, \eta_n)$
c1. $A_\sigma > 1, \lambda^* = \mu_0$	Always	$f_\sigma((\mu_0, H))$
c2. $A_\sigma > 1, \lambda^* < \mu_0$	$(\mu_0, \infty) \subseteq \text{dom}(f_\sigma)$	$\bigcup_{n \geq 0} (\mu_n, \eta_n)$

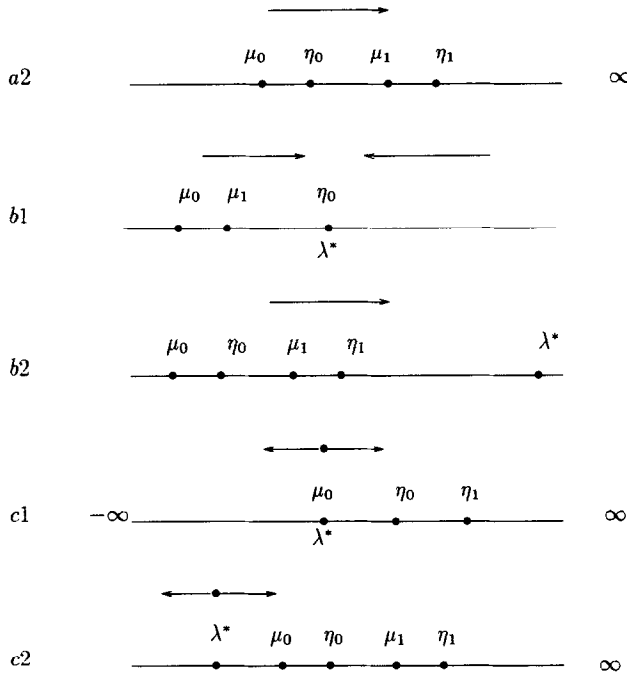


Fig. 12. An illustration of cases a2–c2 of Table 3.

The cases

$$A_\sigma = 1, B_\sigma < 0, \quad A_\sigma < 1, \lambda^* < \mu_0, \quad A_\sigma > 1, \eta_0 < \lambda^*$$

are treated symmetrically to the cases

$$A_\sigma = 1, B_\sigma > 0, \quad A_\sigma < 1, \eta_0 < \lambda^*, \quad A_\sigma > 1, \lambda^* < \mu_0.$$

The cases $A_\sigma \neq 1, \mu_0 < \lambda^* < \eta_0$ are treated by splitting the cyclic successor-chain

$$(e_1, (l_1, h_1)), \dots, (e_k, (l_k, h_k))$$

into the two chains

$$(e_1, (l_1, \lambda^*)), \dots, (e_k, (l_k, \lambda^*)) \quad \text{and} \quad (e_1, (\lambda^*, h_1)), \dots, (e_k, (\lambda^*, h_k))$$

and considering each of them separately.

Without loss of generality, we may assume that $e' = e_1$, i.e., that we wish to check for the reachability of an interval which lies on the initial edge e_1 . To check that some point in $(e', (l', h'))$ is reachable, we have to check whether (l', h') intersects the interval (μ_0, η_0) (cases a1 and b1) or the interval $f_\sigma(\mu_0, H)$ (case c1). For cases a2–c2, it is necessary to check whether (l', h') intersects any of the intervals (μ_n, η_n) , for some $n \geq 0$. Since, in all three cases, the two sequences μ_0, μ_1, \dots and η_0, η_1, \dots are monotonically increasing, and have closed-form expressions, it is straightforward to check for the necessary intersection. \square

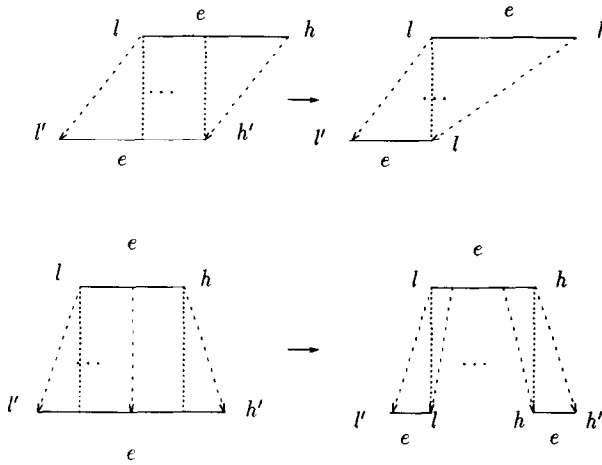


Fig. 13. Transforming a successor-tree (left) into a nonredundant one (right): the nonsplit case (up) and the split case (down).

Definition 10 (*Non-redundant successor-tree*). A nonredundant successor-tree is obtained from a successor-tree via the following recursive transformation: if $(e, (l', h'))$ is a cyclic successor of $(e, (l, h))$ then replace it by $(e, (l', h')) - (e, (l, h))$ (Fig. 13).

Usually, the modified successor will be a single interval, but in the case of $A_\sigma > 1$ and $l < B_\sigma/(1 - A_\sigma) < h$ the result might be two nonconnected intervals of the same edge – see Fig. 5. Clearly, in terms of *reachable states* the nonredundant tree carries the same information as the original tree, namely the union of the edges is still the set of reachable boundary points.

A node in a tree is called a *branching node* if it has more than one successor.

Claim 10 (*Finitely many chains*). Every infinite chain in a nonredundant successor-tree, has only finitely many branching nodes. Hence this tree contains only finitely many chains.

Proof. Since all infinite chains are ultimately periodic it suffices to consider the periodic part of such chains. Let $(e_1, (l_1, h_1)), (e_2, (l_2, h_2)), \dots$ be an infinite successor-chain with a periodic signature σ^ω , $\sigma = (e_1, \dots, e_{k-1})$. Let $(\mu_0, \eta_0), (\mu_1, \eta_1), \dots$ denote the intervals in the successor-chain that correspond to successive visits at the edge e_1 , starting at the second. Thus, $(\mu_0, \eta_0) = (l_k, h_k), (\mu_1, \eta_1) = (l_{2k}, h_{2k})$, etc.

We consider each of the six cases of Table 3 and show that each of them contains only finitely many branching nodes, under the infinite extensibility condition. Note that the chains we consider here obey the nonredundance condition. In particular, the intervals (μ_i, η_i) and (μ_j, η_j) are disjoint for every $i \neq j$. Also note that branching can arise between the nodes $(e_1, (\mu_i, \eta_i))$ and $(e_1, (\mu_{i+1}, \eta_{i+1}))$ only if there exists some $\lambda \in (\mu_i, \eta_i)$ such that $f_\sigma(\lambda)$ is undefined.

We consider several cases together.

Case a1: $A_\sigma = 1$, $B_\sigma = 0$ and case b1: $A_\sigma < 1$, $\lambda^* = \eta_0$.

In these two cases, $f_\sigma(\lambda)$ is defined for every $\lambda \in (\mu_0, \eta_0)$ and $f_\sigma((\mu_0, \eta_0)) \subseteq (\mu_0, \eta_0)$. Consequently, all the e_1 -points reachable after one round of σ are redundant and the nonredundant chain cannot be infinite.

Case a2: $A_\sigma = 1$, $B_\sigma > 0$, case b2: $A_\sigma < 1$, $\eta_0 < \lambda^*$, and case c2: $A_\sigma > 1$, $\lambda^* < \mu_0$.

It can be shown that, under the infinite extensibility condition, $f_\sigma(\lambda)$ is defined for every $\lambda \in (\mu_k, \eta_k)$ and $(\mu_{k+1}, \eta_{k+1}) \subseteq f_\sigma((\mu_k, \eta_k))$, for every $k \geq 0$. It follows that this chain cannot have a branching node following the second visit to e_1 .

Case c1: $A_\sigma > 1$, $\lambda^* = \mu_0$.

As $\mu_0 < \eta_0$, there is a sufficiently small $\varepsilon > 0$ such that $f_\sigma((\mu_0, \eta_0 + \varepsilon)) \subseteq (\mu_0, \eta_0)$. By nonredundancy, this implies that $\lambda^* < \mu_1$ which shows that from the second σ -round on, this case behaves like case c2: $A_\sigma > 1$, $\lambda^* < \mu_0$.

It follows that every chain in the nonredundant tree has finitely many branching points. By a König-like argument, this implies that the tree has only finitely many chains, some of which may still be infinite. \square

Consequently we have the following theorem.

Theorem 11 (Edge-to-edge reachability). *For every deterministic planar PCD system it is decidable whether an edge interval $(e', (l', h'))$ is reachable from an edge interval $(e, (l, h))$.*

Proof. The algorithm develops top-down the non-redundant successor-tree starting at $(e, (l, h))$ where along every chain it keeps track of unabandoned ancestors. Using this information we can detect edge-cycles and calculate their successor functions and limits. All the infinite chains are ultimately periodic and there are only finitely many of them. Once, it is realized, using the criteria of Claims 9 and 10, that the currently examined chain has a non-branching infinite periodic nonredundant continuation, we need not expand it any longer, but can use the methods of Claim 9 to decide whether it ever intersects $(e', (l', h'))$. \square

As an immediate result we have the following corollary.

Corollary 12 (Region-to-region reachability). *Let P and P' be two finite unions of polygonal sets. Then the problem $\mathbf{Reach}(\mathcal{H}, P, P')$ is decidable.*

Proof. Inside the regions we have two-sided determinism so every region-to-region reachability problem can be reduced to a finite number of edge-to-edge reachability problems. \square

6. Undecidability for three dimensions

The particular topological properties of the plane played a major role in the convergence of our decision procedure. In this section we show that one additional dimension renders the reachability problem undecidable. This negative result will be based on demonstrating the computational power of PCD systems. We will show that three-dimensional PCD systems can simulate, in a sense described below, arbitrary Turing machines.

A *deterministic transition system* is $\mathcal{A} = (Q, \delta)$, where Q is a countable set of states and $\delta: Q \rightarrow Q$ is the transition function. A *run* of \mathcal{A} starting at some q is a (finite or infinite) sequence $\tau = q[0], \dots$, where $q[0] = q$ and $q[i] = \delta(q[i-1])$ for every $i > 0$. The set of all such sequences (*runs*) starting at some $q \in Q_0 \subseteq Q$ is denoted by $L(\mathcal{A}, Q_0)$. We use $\delta^i(q)$ to denote $\delta(\delta^{i-1}(q))$ with $\delta^1(q) = \delta(q)$.

Definition 11 (*Simulation*). Let $\mathcal{A} = (Q, \delta)$ be a deterministic transition system and let Q_0 be a subset of Q . We say that \mathcal{A} is Q_0 -simulated by a PCD system $\mathcal{H} = (X, f)$ if there exists a subset Y of X and a bijection $\pi: Y \rightarrow Q_0$ (state-mapping) such that for every $x, x' \in Y$, there is a trajectory in \mathcal{H} from x to x' (not intersecting Y except in x and x') iff for some i , $\delta^i(\pi(x)) = \pi(x')$ and for every j , $0 < j < i$, $\delta^j(\pi(x)) \notin Q_0$.

By transitivity a trajectory between any x and x' in Y exists iff there is a run $\tau = q[0], \dots, q[m] \in L(\mathcal{A}, Q_0)$ such that $\pi(x) = q[0]$ and $\pi(x') = q[m]$. When $Q_0 = Q$ we say simply that \mathcal{H} simulates \mathcal{A} and in this case an algorithm for solving the reachability problem for \mathcal{H} solves the reachability (and in particular, the halting) problem for \mathcal{A} .

Remark. There is a variety of other notions of simulation between discrete and continuous dynamical systems, but these semantical issues are subject of an independent ongoing research (see [4, 5]). The simple definition we use here is sufficient for the purpose of proving undecidability.

6.1. Simulation of finite-state machines

Here we show how every finite-state automaton can be simulated by a three-dimensional PCD system. A finite automaton without input is a rather trivial object and the construction is presented here just because it underlies the more complicated constructions for infinite-state machines.

Claim 13 (*Simulation of finite automata*). *Every finite deterministic automaton can be simulated by a three-dimensional PCD system.*

Proof. Suppose the automaton has n states. The simulating system is defined over the subset $[1, n] \times [0, 1] \times [0, n]$ of \mathbb{R}^3 . It consists of the regions (we call the state variables x, y , and z) shown in Table 4.

The regions U_{ij} and B_{ij} are defined for every i, j such that $\delta(q_i) = q_j$. As a state-mapping we take $\pi(x, y, z) = q_i$ iff $(x = i) \wedge (y = z = 0)$. An example of this construction appears in Fig. 14. It can be verified that the system goes from $(i, 0, 0)$ to $(j, 0, 0)$ iff $\delta(q_i) = q_j$. At F the system advances y until $(i, 1, 0)$, then at U_{ij} it goes “up” until it reaches the surface $z = j$ at $(i, 1, j)$. On that surface in region B_{ij} it goes diagonally to $(j, 0, j)$ and finally in D it goes down to $(j, 0, 0)$. Note that all the regions leading to $(j, 0, j)$ are located on the same plane. \square

Remark. This technique can be applied to the simulation of nondeterministic automata by *nondeterministic* PCD systems. All we have to do is to modify the definition of a PCD system to allow nondeterminism on the (relative boundaries of the regions. Then if both (q_i, q_j) and (q_i, q_k) are possible transitions, $j < k$, the system will bifurcate in $(i, 1, j)$ between B_{ij} (going to q_j) and U_{ik} (going up until $z = k$ and then to B_{ik}). In [4] we have shown that deterministic PCD systems can simulate (in a richer sense) nondeterministic automata.

Table 4

Region	Defining conditions	$c = (\dot{x}, \dot{y}, \dot{z})$
F	$(z = 0) \wedge (y < 1)$	$(0, 1, 0)$
U_{ij}	$(x = i) \wedge (y = 1) \wedge (z < j)$	$(0, 0, 1)$
B_{ij}	$(z = j) \wedge (x + (j - i)y = j) \wedge (y > 0)$	$(j - i, -1, 0)$
D	$(z > 0) \wedge (y = 0)$	$(0, 0, -1)$

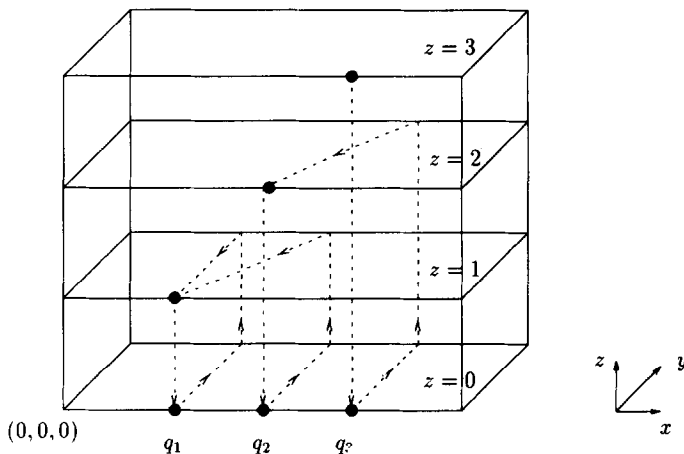


Fig. 14. Simulating a three-state automaton with $\delta(q_1) = \delta(q_2) = q_1$ and $\delta(q_3) = q_2$.

6.2. Pushdown automata

The results concerning infinite-state transition-systems will be based on stack machines. A pushdown stack is an element of Σ^*0^ω , where³ $\Sigma = \{0, \dots, k-1\}$. We define the following two functions: $\text{PUSH}: \Sigma \times \Sigma^\omega \rightarrow \Sigma^\omega$ and $\text{POP}: \Sigma^\omega \rightarrow \Sigma \times \Sigma^\omega$ as $\text{PUSH}(v, S) = v \cdot S$ and $\text{POP}(v \cdot S) = (v, S)$.

Definition 12 (PDAs). A deterministic pushdown automaton (PDA) is a transition system $\mathcal{A} = (Q \times \Sigma^*0^\omega, \delta)$ for some $Q = \{q_1, \dots, q_n\}$ such that δ is defined using a finite collection of statements of one of the following two forms:

$$\begin{array}{ll} q_i: & S := \text{PUSH}(v, S); \\ & \text{GOTO } q_j \\ & \dots \\ & \text{IF } v = 0 \text{ GOTO } q_{i_0}; \\ & \dots \\ & \text{IF } v = k-1 \text{ GOTO } q_{i_{k-1}}; \end{array}$$

The contents of a stack is denoted by $S = s_1 s_2 \dots$, where s_1 is the top of the stack. We define the standard encoding function $r: \Sigma^\omega \rightarrow [0, 1]$ as $r(S) = \sum_{i=1}^{\infty} s_i k^{-i}$. Clearly r is injective on Σ^*0^ω . It is easily verified that the stack operations have arithmetic counterparts that operate on the representation:

$$\begin{aligned} S' = \text{PUSH}(v, S) & \text{ iff } r(S') = (r(S) + v)/k, \\ (S', v) = \text{POP}(S) & \text{ iff } r(S') = kr(S) - v. \end{aligned}$$

Claim 14 (Simulation of PDAs). *Every PDA can be simulated by a three-dimensional PCD system.*

Proof. For simplicity we assume $k = 2$ and $\Sigma = \{0, 1\}$. Consider the three planar subsystems depicted in Fig. 15 and a trajectory segment starting at $x = (x, 0)$, $x \in [0, 1]$ and ending at $x' = (x', 1)$. It can be verified that either

$$\begin{aligned} x' &= (x + 1)/2 \quad (\text{PUSH } 1), \\ x' &= x/2 \quad (\text{PUSH } 0), \\ x' &= 2x - 1/2 \quad (\text{POP}). \end{aligned}$$

If $x = r(S)$ at the “input port” ($y = 0$) of a PUSH element, then $x' = r(S')$ at the “output port” ($y = 1$) of that element where S' is the resulting stack. For the POP element we have two output ports $-\frac{1}{2} \leq x < \frac{1}{2}$ and $\frac{1}{2} \leq x < \frac{3}{2}$. If the top of the stack was 0 the trajectory reaches the left port with $x' = r(S') - \frac{1}{2}$, otherwise it goes to the

³ It is more convenient to define the set of all stacks as a countable subset of Σ^ω although it is isomorphic to Σ^* .

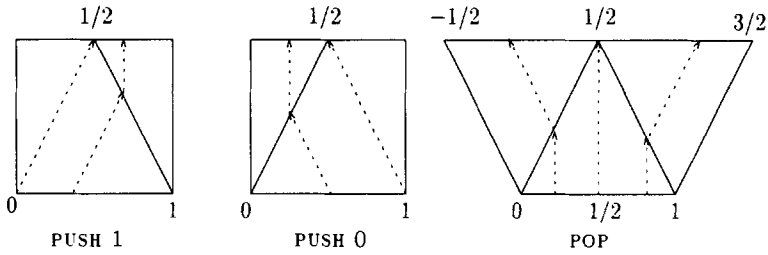


Fig. 15. The basic elements.

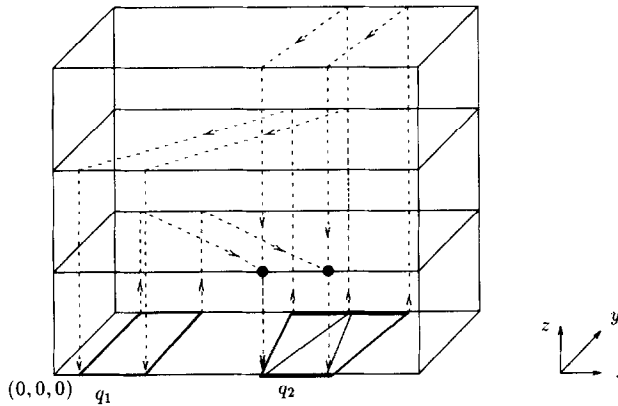


Fig. 16. Simulating a PDA with two states, defined by: $q_1: S := \text{PUSH}(1, S); \text{GOTO } q_2; q_2: (v, S) := \text{POP}(S);$ If $v = 1$ THEN GOTO q_2 ELSE GOTO q_1 . Note the place where the two GOTOs to q_1 merge.

right port with $x' = r(S') + \frac{1}{2}$. In both cases the value of x' (relative to the “origin” of the port) encodes the new content of the stack. Thus, in order to simulate a PDA we pick for every q_i an element corresponding to its stack operation, place it with the origin in position, say, $(2i, 0, 0)$ and use the third dimension in order to connect the output ports back to the input ports according to the GOTO’s (see Fig. 16). This is similar to the previous construction except for the fact that the connections are via two-dimensional “bands” and thus two families of trajectories going to the same state q_i cannot be merged on the same plane ($z = j$) but only while going “down”. Finally the state-mapping is defined as $\pi(x, y, z) = (q_i, S)$ iff $y = z = 0$, $2i \leq x < 2i + 1$ and $S = r^{-1}(x - 2i)$. \square

6.3. Simulating two-stack and Turing machines

The construction of Claim 14 generalizes naturally to automata having two stacks (2PDAs). We can define an encoding function $\bar{r}: \Sigma^\omega \times \Sigma^\omega \rightarrow [0, 1] \times [0, 1]$ by letting $\bar{r}(S_1, S_2) = (r(S_1), r(S_2))$. This way every configuration of the two stacks can be encoded by a point $x = (x_1, x_2, 0)$ in a two-dimensional input port. The elements that

simulate the stack operations $\text{PUSH}(v, S_1)$, $\text{PUSH}(v, S_2)$, $\text{POP}(S_1)$ and $\text{POP}(S_2)$ operate on the appropriate dimension, according to the stack involved, and leave the other dimension intact. As an example, an element corresponding to $\text{PUSH}(0, S_1)$ appears in Fig. 17. From this we can immediately conclude:

Claim 15. *Every 2PDA (and hence any Turing machine) can be simulated by a four-dimensional PCD system.*

Proof. As in Claim 14 we pick n elements, arrange them along a line and connect output ports to input ports. The connections should now “carry” two-dimensional information about the configuration, and thus consist of three-dimensional “tubes”. Several tubes going to the same state can be merged by employing a fourth dimension (no figure) in the same way as two-dimensional “bands” were merged in the case of one-stack PDAs. \square

But we can do better. First, we impose some restrictions on the type of 2PDAs used.

Definition 13 (Normal 2PDA). Let $\Sigma = \{0, 1, 2\}$. A configuration $(S_1, S_2) \in \Sigma^\omega \times \Sigma^\omega$ is normal if both S_1 and S_2 belong to $\{1, 2\}^* 0^\omega$. A 2PDA is normal if it never pushes 0 to any of the stacks.

It can easily be verified that normality of the configurations is preserved by normal 2PDAs and that normal 2PDAs can simulate Turing machines (see the proof of equivalence of Turing machines and 2PDAs, e.g., [11]).

Definition 14. Let $\mathcal{A} = (Q \times \Sigma^* 0^\omega \times \Sigma^* 0^\omega, \delta)$ be a 2PDA and let C_0 be a set of configurations. With every $q_i, q_j \in Q$ we associate the set

$$S(C_0, i, j) = \{(S_1, S_2) \in \Sigma^\omega \times \Sigma^\omega : (\exists \tau \in L(\mathcal{A}, C_0)) (\exists k > 0) \\ \tau[k] = (q_i, S'_1, S'_2) \wedge \tau[k+1] = (q_j, S_1, S_2)\}$$

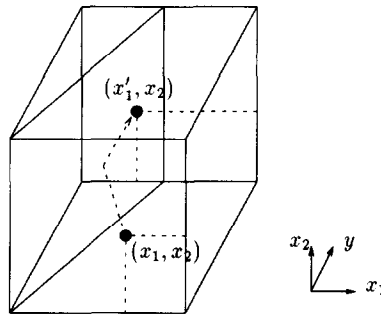


Fig. 17. An element simulating the operation $\text{PUSH}(0, S_1)$.

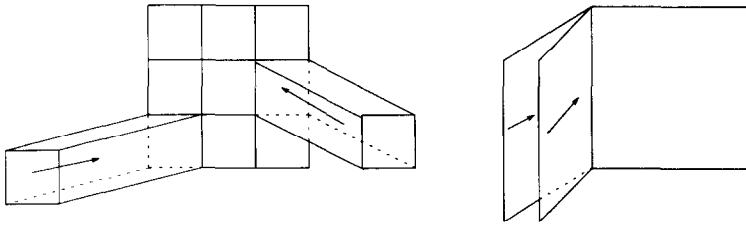


Fig. 18. Realizing entrances to input ports of separated (left) and flat (right) states.

In other words, $S(C_0, i, j)$ is the set of all 2-stack configurations with which a transition from q_i to q_j can take place in any run of \mathcal{A} starting at C_0 .

Definition 15 (*Separated and flat states, regular 2PDAs*). A state q_k of a 2PDA is separated if for every $q_i \neq q_j \in Q$, if both q_i and q_j have a GOTO q_k instructions, they are preceded, respectively, by PUSH v_i and PUSH v_j (into the same stack) for $v_i \neq v_j$. A state q_k of a 2PDA is C_0 -flat if for every j , $S(C_0, j, k) \subseteq \{0^\omega\} \times \Sigma^\omega$. A 2PDA is C_0 -regular if each of its states is either separated or C_0 -flat.

A state q_k is thus separated if, upon entering q_k , the values on the tops of the stacks are sufficient to tell whether we come from q_i or q_j . A state q_k is C_0 -flat if in all the runs starting at C_0 it is always entered with the first stack empty.

Claim 16 (Simulation of regular 2PDAs). *Let C_0 be a set of configurations. Any C_0 -regular 2PDA can be C_0 -simulated by a three-dimensional PCD system.*

Proof. In the proof of Claim 15 we needed the fourth dimension only in order to merge two or more “tubes” entering the same input port associated with a state q . If q is separated these tubes do not overlap on the input port and the connections can be made. If q is flat, the relevant information at the input port of q is one-dimensional and all incoming “bands” can be glued together (see Fig. 18).

Note that the relativity of flatness and of simulation with respect to C_0 is important. A trajectory starting at some configuration outside C_0 might want to enter the input port of q_k at some point with $x_1 \neq 0$, but since we are only interested in C_0 -simulation we do not care.

What we are going to show is, informally speaking, that every normal 2PDA can be transformed into an N -regular 2PDA, where N is the set of normal configurations. The idea is simple: each time after performing a stack operation, we empty one stack while pushing its contents into the other. Then we perform the GOTO’s, i.e. merge several “bands” that contain one-dimensional information. Before entering the new input port we decode the one-dimensional representation back into two stacks.

For every $i, j \in \{1, \dots, n\}$, we define a machine $Encoder_{ij}$ and a machine $Decoder_j$. An encoder takes two normal stack configurations $S_1 = \alpha_1, \dots, \alpha_l 0^\omega$ and $S_2 = \beta_1, \dots, \beta_m 0^\omega$ ($\alpha_i, \beta_i \in \{1, 2\}$) and converts them into $S_1 = 0^\omega$ (“empty” stack) and

$S_2 = \alpha_1 \dots \alpha_1 0 \beta_1 \dots \beta_m 0^\omega$. The decoder does the reverse operation. These two machines are described below:

Encoder _{ij}	Decoder _j
E-Entry _{ij} : $S_2 := \text{PUSH}(0, S_2)$ GOTO E-Loop _{ij}	D-Entry _j : $S_1 := \text{PUSH}(0, S_1)$ GOTO D-Loop _j
E-Loop _{ij} : $(S_1, v) := \text{POP}(S_1)$ IF $v = 0$ GOTO E-Exit _{ij} IF $v = 1$ GOTO E-Mov1 _{ij} IF $v = 2$ GOTO E-Mov2 _{ij}	D-Loop _j : $(S_2, v) := \text{POP}(S_2)$ IF $v = 0$ GOTO D-Exit _j IF $v = 1$ GOTO D-Mov1 _j IF $v = 2$ GOTO D-Mov2 _j
E-Mov1 _{ij} : $\text{PUSH}(1, S_2)$ GOTO E-Loop _{ij}	D-Mov1 _j : $\text{PUSH}(1, S_1)$ GOTO D-Loop _j
E-Mov2 _{ij} : $\text{PUSH}(2, S_2)$ GOTO E-Loop _{ij}	D-Mov2 _j : $\text{PUSH}(2, S_1)$ GOTO D-Loop _j
E-Exit _{ij} : GOTO D-Entry _j	D-Exit _j : GOTO q_j

Claim 17 (Normal \Rightarrow N-regular). *Let $\mathcal{A} = (Q \times \Sigma^* 0^\omega \times \Sigma^* 0^\omega, \delta)$ be a normal 2PDA and let N be the set of all normal configurations. Then there is an N-regular 2PDA $\mathcal{A}' = ((Q \cup Q') \times \Sigma^* 0^\omega \times \Sigma^* 0^\omega, \delta')$ such that for every $q, q' \in Q$ and every $(S_1, S_2), (S'_1, S'_2) \in N$ there is a run from (q, S_1, S_2) to (q', S'_1, S'_2) in \mathcal{A} iff there is such a run in \mathcal{A}' .*

Proof. We let Q' be the union of the set of states of the corresponding encoders and decoders. The transition function δ' is the union of the transitions of the encoders and decoders and the following variation of δ : every original \mathcal{A} -statement of the form $q_i: \dots \text{GOTO } E\text{-Entry}_{ij}$ is replaced by $q_i: \dots \text{GOTO } E\text{-Entry}_{ij}$. It can easily be verified that every q_j is now separated (it is entered only from $D\text{-Exit}_j$), that $E\text{-Entry}_{ij}$ is separated (it is entered only from q_i) and that $D\text{-Entry}_j$ is N -flat (all trajectories starting with a normal configuration will enter the encoder with a normal configuration and will leave the encoder with one stack empty). The other states of the decoders and encoders are obviously separated. This construction is drawn schematically in Fig. 19.

Theorem 18 (Simulation of 2PDAs). *Any normal 2PDA \mathcal{A} can be simulated by a three-dimensional PCD system.*

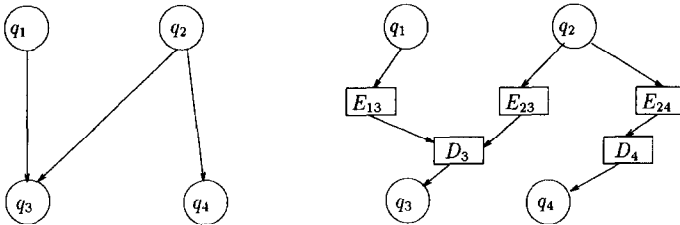


Fig. 19. Augmenting a 2PDA with encoders and decoders.

Proof. We convert \mathcal{A} into the N -regular 2PDA \mathcal{A}' described above. This 2PDA can be N -simulated in three dimensions. By letting $\pi(x, y, z) = (q_i, S_1, S_2)$ iff $y = 0$, $2i \leq x < 2i + 1$, $S_1 = r^{-1}(x - 2i)$, $S_2 = r^{-1}(z)$ and both S_1 and S_2 are normal, we obtain the desired simulation. \square

Corollary 19 (Undecidability). *The reachability problem for three-dimensional PCD systems is undecidable.*

Proof. Otherwise we could translate every reachability problem of a Turing machine, into a reachability problem between two rational points in a three-dimensional simulating system, and solve the halting problem. \square

7. Discussion

We have defined PCD systems, a class of simple dynamical systems, which is a natural extension of real-time systems (such as timed automata or integration graphs) whose trajectories can be computed effectively and precisely. The class of behaviors exhibited by PCD systems is rather rich even without exploiting the full power of the HS model. We have utilized the constraints on behavior imposed by the topological structure of the plane in order to devise a decision procedure for the reachability problem in the two-dimensional case. We have shown that in higher dimensions, when these properties do not hold anymore, the problem becomes undecidable.

The systems constructed for the undecidability results do not seem to be “naturally occurring” and an open question that remains is what additional restrictions on a PCD system will make the reachability problem decidable regardless of dimensionality. Even for the undecidable cases, our techniques can serve as a basis for semi-decidable symbolic simulation techniques as advocated in [15]. The idea is to start with a formula (= a region of possible initial conditions) and calculate successively formulas that characterize reachable states. Then, if the process converges, a test for intersection between the region expressed by the obtained formula and the target region should be performed. Our decidability result can be viewed as an “on-the-fly” version of the approach suggested in [15], and it works even in the case where no finite formula over the reals can characterize the reachable states (as in the case of a spiral).

The applicability of these methods to more general hybrid systems, having nonconstant derivatives in every state, is currently under investigation. This general case poses a new dimension of problems associated with the inability to calculate trajectories and solve questions using exact methods and hence the need for numerical approximations.

In addition to the undecidability result we have shown (in [4]) additional interesting connections between topological properties of dynamical systems and their computational expressiveness. For example, two-dimensional PCD systems cannot simulate nondeterministic automata having a nonplanar transition graph.

There have been other works on simulation of transition systems by dynamical systems. For example, in [9] the boolean transition function of an automaton, defined over $\{0, 1\}^m$, is realized by its continuous extension to $[0, 1]^m$ using arithmetical operations. Similarly stacks have been simulated by rational arithmetic in [8]. In these works, however, the simulating system is *already defined over discrete time*, i.e., using iterated maps of the form $x_{n+1} = f(x_n)$. Our construction, on the other hand, uses continuous-time systems.

There have been various undecidability results for other variants of hybrid systems with piecewise-constant derivatives (*timed automata* [3] or *integration graphs* [12]), but those were obtained in a richer model where a transition between regions is accompanied by a discrete change, and the trajectories are discontinuous. In [6] automata were simulated (without a precise formal definition of this term) by smooth dynamical systems defined over a state-space of certain symmetric metrices. Those systems have high dimensionality that grows with the size of the automaton. Recently, Branicky [5] used differential equations with continuous vector fields, as well as several models of hybrid systems, to simulate Turing machines. The dynamical systems considered in [5] have, informally speaking, infinitely many regions (or components) unlike our PCD systems.

The closest work to ours has been reported in [17] where stack machines were constructed from optical elements such as mirrors and lenses. These constructions were used to prove undecidability of the ray tracing problem. It should be noted, however, that optical systems, as well as billiard models, require a richer model, where the phase-space is $2n$ -dimensional (the velocity in each spatial dimensions is also a state variable) and the trajectories are discontinuous in this phase-space (the velocity goes through an abrupt change). Hence the equivalence between our PCD results and theirs is an optical illusion.

Finally the philosopher Putnam [16], while attempting to “prove” the thesis *every open physical system realizes every automaton*, used a notion of simulation we find implausible. Consider, for example, a deterministic automaton without input, generating the sequence $(q_1 q_2)^\omega$. Then the dynamical system $dx/dx = 1$ (or any other system with a noncyclic behavior) simulates the automaton by letting $\pi(x) = q_1$ when $2i < x < 2i + 1$, and $\pi(x) = q_2$ when $2i + 1 < x < 2i + 2$ for any integer $i \geq 0$. This simulation works only if we consider a fixed initial state (otherwise we need a different abstraction for each state) and, moreover, π is topologically rather complex: $\pi^{-1}(q)$ is a union of infinitely many disconnected sets, which contradicts our intuition concerning abstractions.

Acknowledgements

We would like to thank A. Bouajjani, S. Yovine and J. Sifakis for many helpful comments. In particular, A. Bouajjani contributed to an earlier undecidability result for six dimensions that inspired the current construction.

References

- [1] R. Alur, C. Courcoubetis and D.L. Dill, Model checking in dense real time, *Inform. and Comput.* **104** (1993) 2–34.
- [2] R. Alur, C. Courcoubetis, T. Henzinger and P. Ho, Hybrid automata: an algorithmic approach to the specification and analysis of hybrid systems, in: R.L. Grossman, A. Nerode, A. Ravn and H. Rischel, eds., *Hybrid Systems*, Lecture Notes in Computer Science, Vol. 736 (Springer, Berlin, 1993) 209–229.
- [3] R. Alur and D.L. Dill, A theory of time automata, *Theoret. Comput. Sci.* **126** (1994) 183–235.
- [4] A. Asarin, and O. Maler, On some relations between dynamical systems and transition systems, in: S. Abiteboul and E. Shamir, eds. *Proc ICALP '94*, Lecture Notes in Computer Science, Vol. 820 (Springer, Berlin, 1994) 59–72.
- [5] M.S. Branicky, Universal computation and other capabilities of hybrid and continuous dynamical systems, *Theoret. Comput. Sci.* **138** (1995) 67–100, this issue.
- [6] R.W. Brockett, Smooth dynamical systems which realize arithmetical and logical operations, in: H. Nijmeijer and J.M. Schumacher, eds., *Three Decades of Mathematical Systems Theory*, Lecture Notes in Control and Information Sciences (Springer, Berlin, 1989) 19–30.
- [7] A. Brøndsted, *An Introduction to Convex Polytopes* (Springer, New York, 1983)
- [8] M. Cosnard, M. Garzon and P. Koiran, Computability properties of low-dimensional dynamical systems, in: P. Enjalbert, A. Finkel and K.W. Wagner, eds., *Proc. 10th Ann. Symp. on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, Vol. 665 (Springer, Berlin, 1993) 365–373.
- [9] R.A. DeMillo and R.J. Lipton, Defining software by continuous smooth functions, *IEEE Trans. Software Eng.* **17** (1991) 383–384.
- [10] T. Henzinger, X. Nicollin, J. Sifakis and S. Yovine, Symbolic model-checking for real-time systems, *Inform. and Comput.* **111** (1994) 193–244.
- [11] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, Reading, MA, 1979).
- [12] Y. Kesten, A. Pnueli, J. Sifakis and S. Yovine, Integration graphs: a class of decidable hybrid systems, in: R.L. Grossman, A. Nerode, A. Ravn and H. Rischel, eds., *Hybrid Systems*, Lecture Notes in Computer Science, Vol. 736 (Springer, Berlin 1993) 179–208.
- [13] O. Maler, Z. Manna and A. Pnueli, From timed to hybrid systems, in: J.W. de Bakker, C. Huizing, W.P. de Roever and G. Rozenberg, eds., *Proc. REX Workshop Real Time: Theory in Practice*, Lecture Notes in Computer Science, Vol. 600 (Springer, Berlin, 1992) 447–484.
- [14] O. Maler and A. Pnueli, Reachability analysis of planar multi-linear systems, in: C. Courcoubetis, ed., *Proc. 5th Work-shop on Computer-Aided Verification*, Lecture Notes in Computer Science, Vol. 697 (Springer, Berlin, 1993) 194–209.
- [15] X. Nicollin, A. Olivero, J. Sifakis and S. Yovine, An approach to the description and analysis of hybrid systems, in: R.L. Grossman, A. Nerode, A. Ravn and H. Rischel, eds., *Hybrid Systems*, Lecture Notes in Computer Science, Vol. 736 (Springer, Berlin, 1993) 149–178.
- [16] H. Putnam, *Representation and Reality* (MIT Press, Cambridge, MA, 1988) 121–125.
- [17] J.H. Reif, J.D. Tygar and A. Yoshida, The computability and complexity of optical beam tracing in: *Proc. 31st Ann. Symp. on Foundations of Computer Science*, St. Louis, MO (IEEE Press, New York, 1990) 106–114.